[2pkt.] Zadanie 1.

Szablon rozwiązania: `zad1.py`

Każdy nieskierowany, spójny i acykliczny graf $G = (V, E)$ możemy traktować jako drzewo. Korzeniem tego drzewa może być dowolny wierzchołek $v \in V$. Napisz funkcję `best_root(L)`, która przyjmuje nieskierowany, spójny i acykliczny graf G (reprezentowany w postaci listy sąsiedztwa) i wybiera taki jego wierzchołek, by wysokość zakorzenionego w tym wierzchołku drzewa była możliwie najmniejsza. Jeśli kilka wierzchołków spełnia warunki zadania, funkcja może zwrócić dowolny z nich. Wysokość drzewa definiujemy jako liczbę krawędzi od korzenia do najdalszego liścia. Uzasadnij poprawność zaproponowanego algorytmu i oszacuj jego złożoność obliczeniową.

Funkcja `best_root(L)` powinna zwrócić numer wierzchołka wybranego jako korzeń. Wierzchołki numerujemy od 0. Argumentem `best_root(L)` jest lista postaci:

$$L = [l_0, l_1, \ldots, l_{n-1}],$$

gdzie l_i to lista zawierająca numery wierzchołków będących sąsiadami i–tego wierzchołka. Można przyjąć (bez weryfikacji), że lista opisuje graf spełniający warunki zadania. W szczególności, graf jest spójny, acykliczny, oraz jeśli $a \in l_b$ to $b \in l_a$ (graf jest nieskierowany). Nagłówek funkcji powinien mieć postać:

```
def best_root(L):
  ...
```

Przykład. Dla listy sąsiedztwa postaci:

```
L = [ [ 2 ],
      [ 2 ],
      [ 0, 1, 3],
      [ 2, 4 ],
      [ 3, 5, 6 ],
      [ 4 ],
      [ 4 ] ]
```

funkcja powinna zwrócić wartość 3.

8. Miasto chce pokryć park kopułami antysmogowymi. Park ma kształt prostokąta podzielonego na T odcinków jednakowej długości. Firma produkująca kopuły ma do dyspozycji określone produkty, dane jako trójki (a_i,b_i,c_i) dla i-tej kopuły, gdzie a i b to końce kopuł (przy czym $a \leq b$), a c to koszt danej kopuły. Chcemy poznać koszt (i czy się w ogóle da, mając do dyspozycji dane kopuły) pokrycia wszystkich odcinków parku, przy czym ze względów technicznych kopuły nie mogą na siebie nachodzić. Należy wyznaczyć funkcję $f(x)$, która wyznacza minimalny koszt pokrycia od odcinka 1 do odcinka x oraz podać wzór rekurencyjny tej funkcji.

Uwagi:

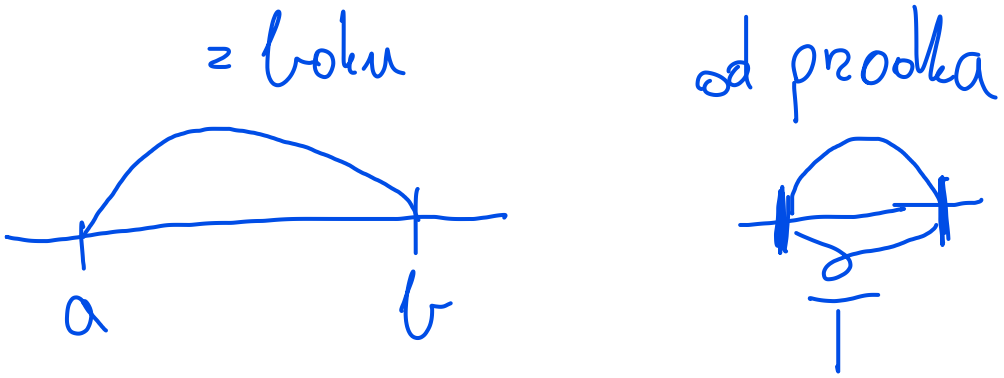
Kazda kopula jest krzywa XD i jej szerokosc jest taka jaka parku .
(mamy pokryć oś OX) do a do b

Wejście:

zawiera tablice P - kopuły, prz - przedział postaci (a,b) - punkt lewy i prawy parku

Wypisz:

-1 jeżeli nie da się pokryć
w przeciwnym przypadku minimalny koszt pokrycia



[2pkt.] Zadanie 3. Niecierpliwy Bob ma wykonać k spośród prac $J_1, \ldots, J_n$, gdzie każda praca jest opisana przez czas rozpoczęcia oraz czas zakończenia:

```
struct Job {
  int start, end; //  czas rozpoczęcia i zakończenia (wyrażone w minutach)
};
```

Bob może wybrać dowolne k prac, byle w jednej chwili nie musiał zajmować się więcej niż jedną. Bob jest niecierpliwy i chce zminimalizować sumę czasu, jaki czeka między wybranymi pracami. Proszę zaimplementować funkcję:

```
int impatientBob( Job J[], int n, int k );
```

która na wejście otrzymuję tablicę n prac (posortowanych rosnąco ze względu na czas zakończenia) i liczbę k, a zwraca minimalną sumę minut, które musi czekać Bob między pracami (lub -1 jeśli nie da się wybrać k niepokrywających się zadań). Proszę skrótowo opisać wykorzystany algorytm.